

Source Code Documentation for Job Partition  
Submitted for LIS Milestone-G

Version 1.1

## Contents

<b>1 Description</b>	<b>2</b>
<b>2 Routine/Function Prologues</b>	<b>2</b>
2.0.1 split1km – program to split the global 1km grid space into sub-domains (Source File: split-1kmblocks.F90) . . . . .	2

## 1 Description

The global spatial-domain is evenly divided into rectangular subdomains. The size of a subdomain can be controlled by the user, with consideration of the compute nodes' capacity and efficient resource utilization. The optimal sizes should be as big as possible to reduce overhead while small enough to fit in a compute node's memory. For example, LIS' 1-km simulation is based on a partition scheme which divides the global domain into 50 X 50 subdomains, with each subdomin having 720 X 300 grid points, as shown in Fig. 1. Some of the subdomains (black boxes) thus obtained may not contain any land points at all, and they will not be processed. Only the subdomains (green boxes) that contain at least one land point will be submitted to the job management system, for the obvious benefit of efficiency.

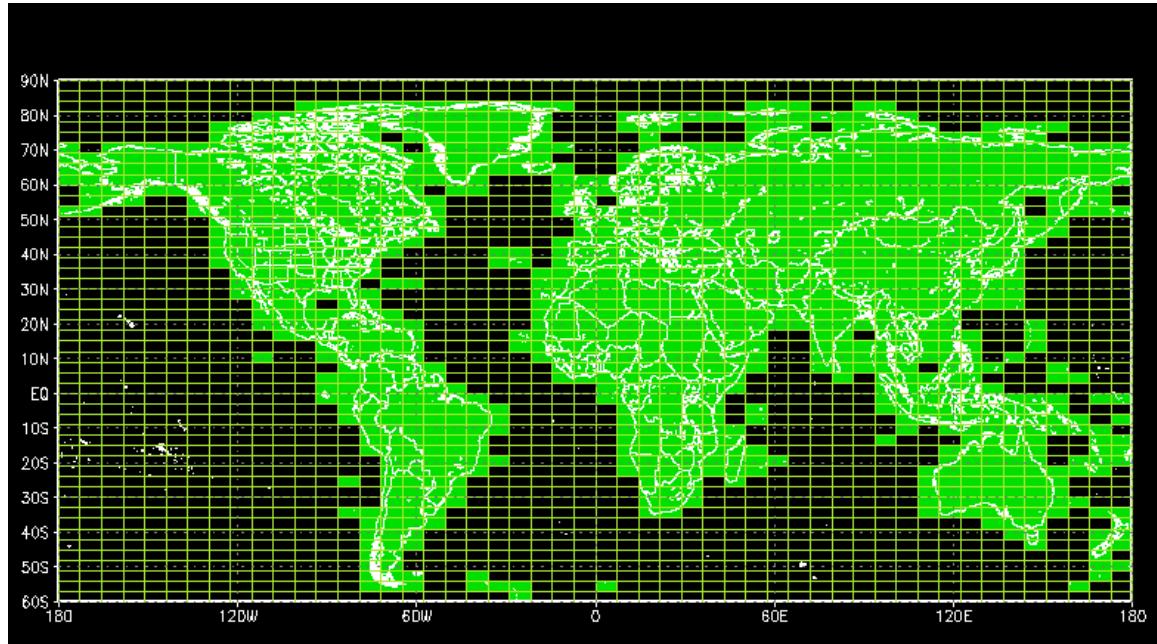


Figure 1: Domain partition for 1km global domain.

## 2 Routine/Function Prologues

### 2.0.1 **split1km – program to split the global 1km grid space into sub-domains** (Source File: *split-1kmblocks.F90*)

This program splits the LIS 1km grid space ( $36000 \times 15000$ ) into  $50 \times 50$  subdomains of  $720 \times 300$  for parallel processing. It counts the number of land points in each subdomain (block) based on the land/sea mask data, so the subdomains with land points will be processed and those without any land points can be discarded.

Each subdomain is identified by its index ( $i_c, i_r$ ) in the longitude and latitude direction on the global grid space.

#### REVISION HISTORY:

*split-1kmblocks.F90*, v 1.1 2004/03/29 20:46:18 Yudong Tian

## CONTENTS:

```

PROGRAM split1km
IMPLICIT NONE
INTEGER X,Y,COL,ROW,J,I,RERR,IR,IC,V,SP4,NP4
integer k,kk,ii,ii1,ii2,jj,elr,elc

!* NOTE: (1,1) is in the lower left corner of the world

!** output grid
REAL, PARAMETER :: XRES=1.0/100.0 !X J E-W Col Lon Resolution in Degrees
REAL, PARAMETER :: YRES=1.0/100.0 !Y I N-S Row Lat Resolution in Degrees
REAL, PARAMETER :: CLON=-180.0 + XRES/2 !Center Lon of (1,1)
REAL, PARAMETER :: CLAT=-60.0 + YRES/2 !Center Lat of (1,1)
INTEGER, PARAMETER :: NC=360/XRES      !Number of columns
INTEGER, PARAMETER :: NR=150/YRES      !Number of rows
INTEGER, PARAMETER :: NX= 50   !Number of columns of blocks
INTEGER, PARAMETER :: NY= 50   !Number of rows of blocks
INTEGER, PARAMETER :: mC=720      !Number of columns in each block
INTEGER, PARAMETER :: mR=300      !Number of rows in each block
INTEGER*1, allocatable :: input(:, :)
INTEGER*1 block(mC, mR)
Integer stat(0:mC*mR) !number of blocks for each number of land points, 0~mc*mr
Integer total

write(*, *) "nc = ", NC, " nr = ", nr
stat = 0
allocate( input(NC, NR), STAT=RERR)
if(RERR.NE.0) then
    Write(*, *) "allocation error: RERR=", RERR
    STOP
END IF
Write(*, *) "Readming mask file ..."
OPEN(11,FILE='UMD_601KMmask.1gd1i',form='unformatted', access='direct', &
recl = NC*NR)
read(11, rec=1, IOSTAT=RERR) input
if(RERR.NE.0) then
    Write(*, *) "allocation error: RERR=", RERR
    STOP
END IF
Close(11)
Write(*, *) "Readming mask file done..."

Open(12, file="land-blocks.txt")
!** get mask
do ir=1, NY
    Write(*, *) "ir = ", ir

```

```
do ic=1, NX
    !** reproject and get the total land points
    total = 0
    Do j = 1, mR
        Do i = 1, mC
            ii = (ic -1) * mC + i
            jj = (ir -1) * mR + j
            If (ii.GT.NC.or.jj.GT.NR) STOP 99
            block(i, j) = input(ii, jj)
            total = total + block(i, j)
        End Do
    End Do
    !buggy "sum" function? total = sum(block)
    stat(total) = stat(total) + 1
    If (total.GT.0) then
        write(12, *) ic, ir, total
    End If
    end do !** ic
end do !** ir
close(12)
Write(*, *) "Writing stat file ..."
Open(14, file="block-stats.txt")
    total = 0 !*** total is total number of land points <= i
    Do i=0, mC*mR
        total = total + stat(i)
        write(14, *) i, total
    End Do
Close(14)

deallocate(input)
Write(*, *) "Writing done..."
```

Stop  
END